UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/802,422	03/17/2004	Jerry Mun Coley	M61.12-0626	3859

27366 7590 10/27/2009

WESTMAN CHAMPLIN (MICROSOFT CORPORATION)
SUITE 1400
900 SECOND AVENUE SOUTH
MINNEAPOLIS, MN 55402

EXAMINER	
VO, TED T	

ART UNIT	PAPER NUMBER
2191	

MAIL DATE	DELIVERY MODE
10/27/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/802,422	COLEY ET AL.	
	Examiner	Art Unit	
	TED T. VO	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *03 August 2009*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,4,12,15 and 26-37* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☒ Claim(s) *1,4,12* is/are allowed.

6)☒ Claim(s) *15, 26-37* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to the amendment filed on 08/03/2009.

Claims 2-3, 5-11, 13-14, 16-25 are canceled.

Claims 1, 4, 12, 15, 26-37 are pending in the application.

Response to Arguments

2. Regarding the arguments' remarks filed on 08/02/09, where in the remarks, Applicants addressed the rejection of claims 29-37. The claims 29-37 have been rejected as they are incorporated in the limitations of the previous claims 1, 3-10, 12-14. Now, claim 1 has been amended and allowed. Claims 29-37 remains rejected and addressed in this office action herein. All the claims will be allowed if they carry out all key limitations in the claim 1.

Applicants argued the claims 15, 26-28. Addressing the argument, the amendment to these claims appears nesting the limitations which are merely general speaking. It could be performed by the .NET IDE in Utley. The combination of Utley and Griffith should address nesting. It appears that the skills of the art will see the references act like a multiple reference since some elements in the claims are inherently used in one reference, but addressed in another reference. For example, 'resource manager', it would be inherently used in Utley but not be shown; it is seen in another reference. Since these claims are amended, the amendment necessitated the new ground of rejection to the claims.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 15, 26-37 are rejected under 35 U.S.C. 103(a) as being unpatentable over Craig Utley, "A Programmer's Introduction to Visual Basic .NET", SAMS Publishing, 2001 (hereinafter: Utley), in view of Griffiths et al (hereinafter: Griffiths), ".NET Windows forms in a Nutshell", April 2003.

As per Claim 15: Utley discloses, ***A computer-implemented method for reducing coding errors prior to runtime in the context of a managed code execution environment*** (See Figure 2.5, p. 27), ***comprising:***

> ***providing a developer with access to a plurality of managed code resources***

in the managed code execution environment, the managed code execution environment including a design program (refer to .NET IDE and FORM in Utley), *a managed code infrastructure* (Set of class in IDE),

and a resource manager (will be combined with Griffith; see "resource manager" addressed in Griffith),

the design program having tools for designing, building, testing, and deploying applications, the managed code infrastructure supporting cross-language compliant code (Utley: p. 16-17), *the*

design program cooperating with managed code infrastructure in production of code that is compliant with a common language specification (Utley: p. 17), *the common language specification comprising a set of basic language features that are common across multiple applications, the set of basic features being exposed in application programming interfaces to other code,* (See Utley, IDE. P. 26; Figure 2.5, p. 27, and Figure 6.15, p. 119)

the resource manager (Inherently from Griffith) *managing resource information for languages supported by the managed code infrastructure, the resource information including information pertaining to resources that are compliant with the common language specification;*

(The description in the recitations of the IDE of Utley, where the IDE in Utley inherently use the resource manager of Griffith);

and verifying that a resource identifier input by the developer corresponds to one of the plurality of managed code resources (Utley: See Figure 2.5, p. 27) ***by:***

providing the developer with a collection of resource identifiers that include at least two identifiers that each identify a different language version of what is essentially the same resource (Utley: See Figure 6.15, p. 119, contains resource identifiers in which the developer recognized CLS-compliant and CLS, as mentioned in appendix A)***; and***

receiving said resource identifier input from the developer in the form of a selection from the collection of resource identifiers (Basis .NET is common language runtime, it provides developers to correspond to managed code resources - see p. 8: 1-7, "The runtime can check to make sure that resources on which you depend are available" - P. 121, Figure 6.15, it has "index", where developers access database to verify the input of visual basic identifiers)

Utley with the .NET IDE does not clearly show Resource manager, but it includes various features, in which it depends on the program type and user choice. Utley shows the .NET IDE with the visually features in the FORM of the .NET; while

Griffiths shows the Resource manager handle at code level, which is inherently in Utley's .NET IDE but not mentioned by Utley.

Thus it is obvious to the ordinary in the art for the combination of the two references to show fully capacity of an IDE, depending upon the types of code and user choice, and the resource manager that provides the insertion of code will be used when it is necessary selected.

As per claim 26: Regarding

26. (Currently Amended) The method of claim 15, wherein verifying comprises utilizing a computer processor that is a functional component of the computer to verify, and

where providing the developer and wherein proving the developer with access to the plurality of managed code resources comprises the develop, During design time interfacing with the design program to create a code that incorporates a call to the resources manager and making, based at least in part of the call, a string resource request to the resource manager utilizing a "GetString (string KeyName)" call.

The claimed recitation is merely general speaking, where the IDE framework inherently provides the developer to access and managed code. See The Figures of Utley; particularly, further see Griffith, sec. Resource Manager (p. 84), the C# of the .NET development provides the user with string resource request using resource manage and GetString.

As per claim 27: Regarding

27. (Currently Amended) The method of claim 26, wherein providing the developer with a collection of resource identifiers comprises displaying the collection of resource identifiers in a pop-up window in a coding area and wherein the design program has an interface that allows the developer to adjust a status of the automatic delivery of resource information from an automatic setting, a resource list automatically appears when an activator key is pressed, wherein for the manual setting, the developer selectively activates the

display of resource information by positioning a cursor at a location associated with information availability.

The claimed recitation is merely general speaking, where the IDE in Utley, for example, Figure 4.4 (p. 87) shows pop-up code area, and with combining the disclose "Resource Manager" Griffith (p. 84) it can provide the developer with a collection of resource identifiers.

As per claim 28: Regarding

28. (Currently Amended) The method of claim 27, further comprising:

automatically inserting a string that corresponds to the selection into a

programming code in the coding area: and

performing a build-time check on the programming code, the build-time check

determining whether resources have been addressed in error, the build-time check comprising resource identifier values in the programming code to a collection of valid values, wherein upon the resource identifier values not matching one of the collection of the values, an error is submitted to the developer for correction.

See every figure in Utley. In the IDE, a FORM builder has tool-bars on the top. Example, see Figure 2.4

As per Claim 29:

Utley discloses,

A computer-implemented method for reducing coding errors prior to runtime in the context of a managed code execution environment, comprising:

receiving from a developer an indication of a desired managed code resource (1)*;*

communicating a resource request to a resource manager (1)*,*

the request including an indication of a key name and a string, the key name and the string both associated with the desired managed code resource (2);

displaying a collection of resource identifiers, each resource identifier (1)

corresponding to the key name and the string; (2)

receiving from the developer a selection that corresponds to one of the resource

identifiers (1) in the collection of resource identifiers (2); and

automatically inserting the one of the resource identifier into a programming code

that is in a coding area (2).

(1) Within the IDE, example, start from p. 30, Figure 2.8, 2-9, 2-10, 2-11, 2-12, etc. The IDE receives any indication via the mouse acting upon a user selection.

(2) Within the IDE as seen in Figure 2.8, 2-9, 2-10, 2-11, 2-12, where as defined, resources and FORM are communicated, where a FORM represents a form class in C# code or VB code (Griffith p. 60) which allows a user to insert code/resources seen/selected from resource field in the IDE. As seen from Griffith, when the user particularly selects a resource in resource files (Discussing in p. 84, "Resource Managers" or page 87-88 "Resource Files", when the Resource manager need to know user particularly selects a resource to insert it in the code area), the IDE provides a selection. User either selects the whole file or strings where the names of resource will be resource entries. The user selection will place his selection into the code that presenting the FORM.

Utley with the .NET IDE does not clearly show Resource manager, but it includes various features, in which it depends on the program type and user choice. Utley shows the .NET IDE with the visually features in the FORM of the .NET; while

Griffiths shows the Resource manager handle at code level, which is inherently in Utley's .NET IDE but not mentioned by Utley.

Thus it is obvious to the ordinary in the art for the combination of the two references to show fully capacity of an IDE, depending upon the types of code and user choice, and the resource manager that provides the insertion of code will be used when it is necessary selected.

As per Claim 30: regarding

The method of claim 29, wherein the indication of a desired
managed code resource comprises a call created in the programming code.

Further see Griffith, For example, in Resource Manger (p. 84) has a call, where within the IDE as seen in Figure 2.8, 2-9, 2-10, 2-11, 2-12, where as defined, resources and FORM are communicated, where a FORM represents a form class in C# code or VB code (Griffith p. 60) which allows a user to insert code/resources seen/selected from resource field in the IDE. As seen from Griffith, when the user particularly selects a resource in resource files (Discussing in p. 84, "Resource Managers" or page 87-88 "Resource Files", when the Resource manager need to know user particularly selects a resource to insert it in the code area), the IDE provides a selection. User either selects the whole file or strings where the names of resource will be resource entries. The user selection will place his selection into the code that presenting the FORM.

As per Claim 31: regarding

The method of claim 30, wherein the call comprises an entry of a resource object followed by an activator key.

The Visual Studio/Basic .NET provides collection of resource identifiers by allowing user to access the database and allows the developers' request and see Griffiths, resource managers, p. 84, the resource manager provide the call. GetString is an activator.

As per Claim 32: regarding

The method of claim 31, wherein the activator key is a period.

In corporate with claim 31, further see in Griffith p. 84, the resource manager provides the call. GetString is an activator, where the key is referred to string and bitmap that are associated with the Form design (See p. 84: lines 1-6). The claim can read on the Visual Studio/Basic .NET which provides the developers to use the cursor and to position at any locations in which the information is available.

As per Claim 33: regarding

The method of claim 31, wherein the activator key is a space bar.

Further see in Griffith p. 84, the resource manager provide the call. GetString is an activator, where the key is referred to string and bitmap that are associated with the Form design (See p. 84: lines 1-6). The claim can read on the Visual Studio/Basic .NET which provides the

developers to use the cursor and to position at any locations in which the information is available.

As per Claim 34: regarding

The method of claim 31, wherein the activator key is a left parenthesis.

Further see in Griffith p. 84, the resource manager provide the call. GetString is an activator, where the key is referred to string and bitmap that are associated with the Form design (See p. 84: lines 1-6). The claim can read on the Visual Studio/Basic .NET which provides the developers to use the cursor and to position at any locations in which the information is available.

As per Claims 35-37: regarding

-The method of claim 31, wherein displaying a collection of resource identifiers comprises displaying the collection in a pop-up window in the coding area.

-The method of claim 35, wherein the pop-up window is located proximate to the call.

-The method of claim 36, wherein stopping on one of the collection of resource identifiers in the pop-up window for a predetermined amount of time causes additional information to appear in a second pop-up window.

Pop-up window is part of the IDE, where though this, IDE/the Visual Studio/Basic .NET, it provides collection of resource identifiers, it allows user to access the database and allows the developers' request. For example, Figure 4.3 (Utley: p. 77), Figure 4.4 (Utley: p. 87). It shows pop-up code area. With combining the teaching "Resource Manager" Griffith (p. 84) it can provide the developer with a collection of resource identifiers. In the showing of resource managers, p. 84 (Griffith), a resource manage which can be associated with the Form design of Utley.

Allowable Subject Matter

5. The following is an examiner's statement of reasons for allowance: Amending claims 1, 12, and 4 overcome the combination of prior arts; Craig Utley, "A Programmer's Introduction to Visual Basic .NET", and Griffiths et at, ".NET Windows forms in a Nutshell".

Conclusion

6. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ted T. Vo whose telephone number is (571) 272-3706. The examiner can normally be reached on 8:00AM to 4:30PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Y. Zhen can be reached on (571) 272-3708.

The facsimile number for the organization where this application or proceeding is assigned is the Central Facsimile number **571-273-8300**.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR

or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

TTV
October 16, 2009

/Ted T. Vo/
Primary Examiner, Art Unit 2191